

TITLE OF INVENTION

Validation of Fusion

CROSS-REFERENCE TO RELATED APPLICATIONS

This invention relates to Provisional Application Serial No. 60/156,676, filed September 29, 1999. The contents of the application are incorporated by reference.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR  
DEVELOPMENT

Not Applicable

REFERENCE TO A MICROFICHE APPENDIX

Not Applicable

BACKGROUND OF THE INVENTION

Function approximation is a term applied to a process of producing a rule that can be used by a computer to figure out what to do when it meets a new set

Cancel

096734-09800









So validation, which is the secondary goal of function approximation, tends to suffer under this method.

The technique for validation of fusion that is presented in this document is an example of validation by inference. A prior technique of validation by inference is described in the paper "Alternative Error Bounds for the Classifier Chosen by Early Stopping", in the Proceedings of the IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing 1997, in Victoria, B.C., Canada, pages 811 to 814, by E. Bax, Z. Cataltepe, and J. Sill, which is hereby incorporated in its entirety by reference. This earlier method of validation by inference is limited to using a single basis function. As a result, the hypothesis function error bound produced by this method is as weak or weaker than the single basis function error bound.

A prior method of validation by inference that uses multiple basis functions is described in the paper "Validation of Voting Committees", in the journal Neural Computation 1998, Volume 10, Issue 4, pages 975 to 986, by E. Bax, which is hereby incorporated in its entirety by reference. This method is limited to classification problems. The method does not apply to problems with continuous-valued functions, known as regression problems.

#### BRIEF SUMMARY OF THE INVENTION

The invention is a technique to validate a hypothesis function, i.e., to eval-







0067734-00000  
122900

data to develop the hypothesis function, the new method generally produces a more accurate hypothesis function. Thus, the new method has an advantage for the primary goal of function approximation – development of a hypothesis function that accurately approximates the target function.

Now compare the new method to the method of using all in-sample data to develop the hypothesis function and then computing a class-based error bound for the hypothesis function. Both methods have access to the same resources to develop the hypothesis function, so neither method has a competitive advantage for the goal of developing an accurate hypothesis function. However, the new method tends to produce a stronger error bound, because the error bound for the new method is based on uniform bounds over a few basis functions instead of uniform bounds over an entire class of prospective hypothesis functions.

The technique for validation by inference presented here also has advantages over previous techniques of validation by inference. The new method uses multiple basis functions. Consequently, it is possible to produce a hypothesis function bound that is stronger than the basis function bounds. Also, the new method applies to regression as well as classification problems.

#### BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

FIG. 1 is a flowchart that illustrates some steps in the process that is the invention; notes in boxes describe process steps, and notes beside arrows describe

data transferred between steps.

## DETAILED DESCRIPTION OF THE INVENTION

### Overview

In the next two subsections, we present the intuition underlying the validation method developed in this paper, and we review the uniform validation technique used to obtain uniform error bounds over the basis functions. In the following subsection, we develop the linear program to infer hypothesis error bounds from basis function error bounds. We show how the method extends beyond fusion to produce error bounds for early stopping and uniform error bounds over multiple hypothesis functions. Then we discuss the intuition behind the linear program. Also, we extend the linear programming technique by showing how to develop additional constraints. Finally, we present test results comparing bounds computed by linear programming to direct error bounds.

### Intuition

We begin with a simple example to demonstrate the process of validation by inference. Consider a single basis function,  $g_1$ , and a single out-of-sample input,  $\mathbf{x}_1$ . Use mean-squared difference as the error function. Let  $f$  be the unknown

**SECRET**

(1)

This example illustrates the following procedure for bounding by inference:

- Now extend the example to include an additional basis function,  $g_2$ . Suppose that we have uniform error bounds for the basis functions:

(2)

11



most distant points from the hypothesis function point  $(0.30, 0.30)$  are distance 0.01 away. Thus, the basis function bounds imply the hypothesis function bound

$$\frac{1}{2}([h(\mathbf{x}_1) - f(\mathbf{x}_1)]^2 + [h(\mathbf{x}_2) - f(\mathbf{x}_2)]^2) \leq 0.005. \quad (4)$$

Note that in this simple example the hypothesis function bound is tighter than the basis function bounds.

Additional basis functions contribute additional constraints, and additional data contribute additional dimensions. In general, the feasible region is the intersection of  $M$   $N$ -dimensional spheres, where  $M$  is the number of basis functions and  $N$  is the number of out-of-sample inputs. Basis functions with lower error and tighter error bounds correspond to smaller spheres and hence to stronger constraints.

To obtain a problem that is computationally tractable, we will use linear programming. We develop the linear program by partitioning the range to discretize the geometric problem of finding a point in the feasible region with greatest distance from the point given by the hypothesis function values on the out-of-sample inputs. In general, the solution of the linear program yields error bounds inferior to those produced by the exact solution of the geometric problem. Subdividing the partitions can produce superior error bounds, but this requires solving a linear program with more variables.

### Review of Uniform Validation



1. The first step is to identify the problem or question that needs to be addressed. This involves understanding the context and the specific requirements of the task.

$$\Pr\{\nu' \geq \nu + \epsilon(t - s)\} \leq e^{-2N\epsilon^2}. \quad (9)$$

For the remainder of this paper, assume  $D = N$ . In practice, if  $D \neq N$ , then substitute (6) for (9).

Let  $\nu_m$  and  $\nu'_m$  represent the validation and out-of-sample error scores for basis function  $g_m$ . Using the sum of probabilities to bound the probability of the union event,

$$\Pr\{\nu'_1 \geq \nu_1 + \epsilon(t-s) \text{ or } \dots \text{ or } \nu'_M \geq \nu_M + \epsilon(t-s)\} \leq M e^{-2N\epsilon^2}. \quad (10)$$

In other words, with probability at least  $1 - Me^{-2N\epsilon^2}$ , for all basis functions, the out-of-sample error score is at most  $\epsilon(t-s)$  greater than the validation error score. Refer to the out-of-sample error bound  $\nu_m + \epsilon(t-s)$  as  $b_m$ .

## Development of the Linear Program

Let  $h$  be the hypothesis function. Partition the range of the target function into  $S_1, \dots, S_K$ . For each out-of-sample input  $\mathbf{x}_n$  and partition  $S_k$ , define

$$y_{nk} = \begin{cases} 1 & \text{if } f(\mathbf{x}_n) \in S_k \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Define  $e_{mnk}$  to be the greatest lower bound on the error score for  $g_m$  if  $f(\mathbf{x}_n) \in S_k$ , i.e.,

$$e_{mnk} = \inf_{v \in S_k} E(g_m(\mathbf{x}_n), v). \quad (12)$$

Similarly, let  $c_{nk}$  be the least upper bound on error score for hypothesis function  $h$  if  $f(\mathbf{x}_n) \in S_k$ , i.e.,

$$c_{nk} = \sup_{v \in S_k} E(h(\mathbf{x}_n), v). \quad (13)$$

The out-of-sample error score for  $h$  is at most

$$\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K c_{nk} y_{nk}. \quad (14)$$

The out-of-sample error bounds  $b_1, \dots, b_M$  for the basis functions imply the constraints:

$$\forall m \in \{1, \dots, M\} \quad \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K e_{mnk} y_{nk} \leq b_m. \quad (15)$$

Hence, the solution of the following integer linear program (ILP) is an upper bound on the out-of-sample error for the hypothesis function.

$$\text{maximize } \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K c_{nk} y_{nk} \quad (16)$$

$$\text{subject to } \forall m \in \{1, \dots, M\} \quad \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K e_{mnk} y_{nk} \leq b_m, \quad (17)$$

$$\forall n \in \{1, \dots, N\} \sum_{k=1}^K y_{nk} = 1, \quad (18)$$

$$\text{and } \forall (n, k) \in \{1, \dots, N\} \times \{1, \dots, K\} \ y_{nk} \in \{0, 1\}. \quad (19)$$

Solving this program may require extensive computation since integer linear programming is NP-complete. To produce a linear program, replace the integer constraints (19) by the constraints

$$\forall(n, k) \in \{1, \dots, N\} \times \{1, \dots, K\} \quad 0 \leq y_{nk} \leq 1. \quad (20)$$



This weakens the constraints, but it ensures that the program has a tractable solution.

### **Use of the Linear Program**

Use the following process to develop basis functions and produce out-of-sample error bounds for a hypothesis function formed by fusion of basis function outputs.

1. Partition a set of in-sample data into a set of training data and a set of validation data.
2. Use the training data and a set of training procedures to develop a set of basis functions.
3. Use the validation data to compute a set of basis function error bounds, with each bound a constraint on the average over a set of out-of-sample inputs of an error function based on basis function outputs and target function outputs. (For example, choose some  $\epsilon > 0$ , and use Formula 10.)
4. Use fusion on the basis functions and the in-sample data to develop a hypothesis function.
5. Form an optimization problem in which:
  - (a) error bounds for the basis functions and the outputs of the basis functions over the out-of-sample inputs are used to form a set of

constraints on the outputs of the target function over the out-of-sample inputs

- (b) and the objective is to bound the average over the set of out-of-sample inputs of an error function based on hypothesis function outputs and target function outputs, subject to the constraints on the target function outputs.

(The optimization problem may be an integer linear program, a linear program, or some other type of optimization problem. For example, the optimization problem may be the integer linear program of Formulas 16, 17, 18, and 19, or the optimization problem may be the linear program of Formulas 16, 17, 18, and 20.)

6. Solve the optimization problem to produce a hypothesis function error bound.

(Note that this process may be implemented using a general purpose computer.) The solution is an out-of-sample error bound for the hypothesis function if the basis function bounds are all valid. For example, by Formula 10, the probability that the basis function bounds are all valid is at least  $1 - Me^{-2N\epsilon^2}$ , where  $M$  is the number of basis functions, and  $N$  is the number of validation examples and the number of out-of-sample examples.

Note that the linear program bound is valid even if the hypothesis function is not the result of fusion of the basis functions. For example, an out-of-sample

error bound for the hypothesis function chosen by early stopping can be computed as follows. Partition the in-sample data into a training set and a validation set. Choose a function at random. Use an iterative method to develop a function that approximates the training data well. Record the sequence of functions produced by the iterations. Sample these functions at intervals to form the set of basis functions. Use the validation data to compute uniform out-of-sample error bounds for the basis functions. Now compute the error over the validation data for every function in the iteration sequence. The function with minimum validation error is the function chosen by early stopping. It is our hypothesis function. With probability at least  $1 - Me^{-2N\epsilon^2}$ , the solution to the linear program is a valid out-of-sample error bound. (For further details, refer to the paper "Alternative Error Bounds for the Classifier Chosen by Early Stopping", which is previously referenced.)

Note that if the uniform basis function bounds are valid, then, for every hypothesis function, the solution of the corresponding linear program is a valid out-of-sample error bound. Uniform basis function bounds logically imply uniform hypothesis function bounds. Thus, we can use linear programs to compute uniform out-of-sample error bounds for any number of hypothesis functions. With probability at least  $1 - Me^{-2N\epsilon^2}$ , the bounds are all valid.

### **Bounds Without Out-of-Sample Inputs**

Briefly, consider the case in which out-of-sample inputs are unknown, but

[illegible]

**SECRET**

**SECRET**

**SECRET**

**SECRET**

**SECRET**

**SECRET**

One strategy to derive additional constraints for the linear program is to validate some aspect of the behavior of the target function over the out-of-sample data, then identify constraints implied by the pattern of behavior. Validating upper bounds on the difference in behavior between the target function and the basis functions (the out-of-sample error) yields the original constraints. Additional constraints can be derived by validating lower bounds on out-of-sample error. Extend the uniform upper bounds for basis function errors in Formula 10 to include lower bounds:

$$\Pr\{\forall m \in \{1, \dots, M\} \mid \nu'_m - \nu_m \mid < \epsilon(t-s)\} \leq 1 - 2Me^{-2N\epsilon^2}. \quad (24)$$

The lower bounds imply the following additional constraints for the linear program.

$$\forall m \in \{1, \dots, M\} \quad \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K e'_{mnk} y_{nk} \geq b'_m, \quad (25)$$

where  $b'_m = \nu_m - \epsilon(t-s)$  is the lower bound corresponding to upper bound  $b_m$ , and  $e'_{mnk}$  is the greatest lower bound corresponding to least upper bound  $e_{mnk}$ .

More constraints can be derived by validating the frequency distribution of the target function output. Recall that the linear program is based on partitioning the range of the target function into subranges  $S_1, \dots, S_K$ . Let  $p_k$  be the fraction of validation inputs for which the target function is in  $S_k$ . Let  $p'_k$  be the (unknown) fraction of out-of-sample inputs for which the target function is in  $S_k$ . Refer to  $p_k$  and  $p'_k$  as placement frequencies. Extend the uniform bounds

for basis function errors to include bounds for placement frequencies:

$$\Pr\{\forall m \in \{1, \dots, M\} \ |\nu'_m - \nu_m| \leq \epsilon(t-s) \text{ and } \forall k \in \{1, \dots, K\} \ |p'_k - p_k| < \epsilon\} \quad (26)$$

$$\leq 1 - 2(M + K)e^{-2N\epsilon^2}. \quad (27)$$

The placement frequency bounds imply the following additional constraints for the linear program.

$$\forall k \in \{1, \dots, K\} \ p_k - \epsilon \leq \frac{1}{N} \sum_{n=1}^N y_{nk} \leq p_k + \epsilon. \quad (28)$$

Other constraints can be derived by validating the rank of the target function output among basis function outputs. For  $m \in \{0, \dots, M\}$ , let  $r_m$  be the fraction of validation inputs for which exactly  $m$  of the basis function outputs are less than or equal to the target function value. Let  $r'_m$  be the corresponding fraction for out-of-sample inputs. Refer to  $r_m$  and  $r'_m$  as rank frequencies. Extend the uniform bounds for basis function errors to include bounds for rank frequencies:

$$\Pr\{\forall m \in \{1, \dots, M\} \ |\nu'_m - \nu_m| \leq \epsilon(t-s) \text{ and } \forall m \in \{0, \dots, M\} \ |r'_m - r_m| \leq \epsilon\} \quad (29)$$

$$\leq 1 - 2(M + M + 1)e^{-2N\epsilon^2}. \quad (30)$$

Use the following partitioning scheme to produce the linear program. Let the number of subranges be one more than the number of basis functions, i.e., let  $K = M + 1$ . For each input, for each  $k \in \{1, \dots, K\}$ , define  $S_k$  to be the

portion of the range containing values greater than exactly  $k - 1$  basis function outputs. This partitioning scheme maintains the original constraints implied by the basis function error bounds, and the rank frequency bounds imply the following additional constraints.

$$\forall k \in \{1, \dots, K\} \quad r_k - \epsilon \leq \frac{1}{N} \sum_{n=1}^N y_{nk} \leq r_k + \epsilon. \quad (31)$$

Within this partitioning framework, there are many frequency distributions that can be validated to produce further constraints. Examples include the rank frequencies for a subset of the basis functions and the frequency with which the target function output is between the outputs of some pair of basis functions. To increase the number of partitions within this framework, subdivide each subrange  $S_k$  into equal-sized sub-subranges, then adjust the constraints accordingly.

Choose the constraints for a given problem without reference to the validation data and the out-of-sample inputs. This restriction ensures the validity of the Hoeffding bounds for uniform validation of the properties that imply the constraints. Choose the nature of the constraints using training data and prior knowledge about the problem. Choose the number of constraints according to the number of validation examples and out-of-sample inputs available. More constraints mean weaker individual constraints, because more constraints require uniform validation of more properties.

## Tests

This section outlines the results of tests on two sets of data. The first data set contains information on credit card users. The second data set contains information on individuals at risk for diabetes. For each data set, the tests compare out-of-sample error bounds produced by linear programming to error bounds produced by applying VC (Vapnik-Chervonenkis) theory directly.

In the first data set, each example corresponds to a credit card user. There are five inputs, corresponding to user traits. A sixth trait is used as the target function. The traits are unknown because the data provider has chosen to keep them secret. The data were obtained from the machine-learning database site at the University of California at Irvine ([www.ics.uci.edu/pub/machine-learning-databases](http://www.ics.uci.edu/pub/machine-learning-databases)). The discrete-valued traits were removed, leaving the six continuous-valued traits. Of the 690 examples in the original database, 24 examples had at least one trait missing. These examples were removed, leaving 666 examples. The data were cleaned by Joseph Sill. The target function values were constrained to the range  $[-1.25, 1.75]$ . In each test, basis function and hypothesis function outputs were also constrained to this range, producing the squared error range  $[s, t] = [0, 9]$ .

In each test, the 666 examples were randomly partitioned into 444 training examples,  $D = 111$  validation examples, and  $N = 111$  out-of-sample examples. In each test,  $M = 10$  basis functions were trained by early stopping. Then a hypothesis function was formed by linear fusion of the basis classifier outputs.



For each basis function, the training data were randomly partitioned into 333 examples used for actual training and 111 examples used for early stopping. The basis functions were artificial neural networks with five input units, five hidden units, and one output unit. The hidden units had tanh activation functions. The initial weights were selected independently and uniformly at random from  $[-0.1, 0.1]$ . Each network was trained by gradient descent on mean squared error over training examples, using sequential mode weight updates with random order of example presentation in each epoch. A snapshot of the weights was recorded after each epoch. The snapshot with minimum error on the 111 early stopping examples was returned as the trained basis function. In each test, the hypothesis function was formed by finding the linear combination of basis function outputs with least squared error over the in-sample data.

Uniform error bounds with 90% confidence were used for both the linear program bound and the direct VC bound. For the linear program, uniform error bounds over the basis functions are attained for  $\epsilon$  small enough that

$$1 - 10e^{-2 \cdot 111 \epsilon^2} \geq 0.90. \quad (32)$$

The best solution is  $\epsilon = 0.144$ . The target function range  $[0, 9]$  was partitioned into  $K = 1000$  intervals to form the linear program.

For the VC bound, we require uniform bounds over the class of 10-dimensional hyperplanes, with 111 validation examples and 111 out-of-sample examples. Using the growth function for linear indicator functions in the book "Statistical Learning Theory", by V. N. Vapnik 1998, which is hereby incorporated in its

$$\epsilon = \sqrt{\frac{-\ln \frac{0.10}{m(2 \cdot 111)}}{2 \cdot 111}}, \quad (33)$$

Table 1 shows the results for each test. Note that the linear program consistently produces out-of-sample error bounds that are tighter than direct VC bounds.

In each test, the 768 examples were randomly partitioned into 368 training examples,  $D = 200$  validation examples, and  $N = 200$  out-of-sample examples. In each test,  $M = 5$  basis functions were trained by early stopping in a similar manner to the tests on credit card data. The training examples were partitioned into 276 for actual training and 92 for early stopping. The basis functions

were networks with seven input units, four hidden units, and a single output unit. The hidden units had tanh activation functions. The initial weights were selected independently and uniformly at random from  $[-20, 20]$ . In each test, the hypothesis classifier was formed by finding the linear combination of basis function outputs with least squared error over the validation data. The error score range was partitioned into  $K = 100$  regions to form the linear program for each test.

Both the linear program error bounds and the direct VC error bounds were based on uniform error bounds with 90% confidence. For the linear program bound,  $\epsilon = 0.099$  achieves this confidence, since there are  $M = 5$  basis classifiers,  $D = 200$  validation examples, and  $N = 200$  out-of-sample examples. For the VC bound,  $\epsilon = 0.244$  is required to have 90% confidence in uniform bounds over the class of five-dimensional linear functions. Table 2 shows the results for each test. Note that the linear program consistently produces tighter out-of-sample error bounds than the direct VC bounds.

In some simple examples in Section 2, validation by inference produces lower out-of-sample error bounds for the hypothesis function than the out-of-sample error bounds for the basis functions. To observe the relationship between hypothesis function and basis function bounds in these tests, the minimum basis function error bound was recorded for each test. Table 3 displays the averages over the 10 tests on each data set of the hypothesis function bound produced by linear programming and the minimum basis function bound. Note that the hy-



Test	Linear Program Out-of-Sample Error Bound	Direct VC Out-of-Sample Error Bound
1	5495	10556
2	5226	10530
3	5868	10515
4	5361	10696
5	5394	10621
6	5690	10642
7	5600	10669
8	5262	10590
9	5372	10557
10	5474	10680
avg	5752	10606

Table 2: Test results for diabetes data. Error bounds are for the hypothesis function, which is a linear combination of the outputs of 5 basis functions. The maximum possible error score is 40000.

Data Set	Hypothesis Function	Basis Function
	Average	Average
	Out-of-Sample	Out-of-Sample
	Error Bound	Error Bound
Credit	2.10	1.92
Diabetes	5752	4823

Table 3: Average out-of-sample error bounds for the hypothesis functions and the basis functions with minimum validation error. The hypothesis function bounds are the results of linear programs based on basis function error bounds. The basis function bounds are the results of uniform validation over basis functions.

Data Set	Hypothesis Function	Basis Function
	Average	Average
	Out-of-Sample	Out-of-Sample
	Error	Error
Credit	0.584	0.605
Diabetes	949	963

Table 4: Average out-of-sample errors for the hypothesis functions resulting from fusion and the basis functions with minimum validation error.

hypothesis function bounds are higher than the minimum basis function bounds. It is not clear how much of the difference in bounds is caused by the underlying geometry of validation by inference, i.e., the differences in behavior between the hypothesis function and the basis functions over the out-of-sample inputs, and how much of the difference is caused by using linear programming to convert the underlying geometric problem into a problem that is computationally tractable.

For these tests, the differences in error bounds show that a lower error bound can be achieved by selecting the basis function with minimum validation error than by developing a hypothesis function through fusion. However, the selected function may not perform as well as the fusion function. Table 4 confirms that the average out-of-sample error is lower for the fusion functions than for the basis functions with minimum validation error. Hence, the methods developed here are appropriate when fusion yields more effective hypothesis functions than selection of a single basis function, and performance has priority over error bounds.